BIOINFORMATICS

Vol. 00 no. 00 2010
Pages 1–6

Savant: Genome Browser for High Throughput Sequencing Data

Marc Fiume [1,*], Vanessa Williams [1], and Michael Brudno [1,2]

[1]Department of Computer Science, University of Toronto, Ontario, Canada
[2]Donnelly Centre and Banting and Best Department of Medical Research, University of Toronto, Ontario, Canada

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Associate Editor: XXXXXXX

ABSTRACT

Motivation: The advent of High Throughput Sequencing (HTS) technologies has made it affordable to sequence many individuals' genomes. Simultaneously the computational analysis of the large volumes of data generated by the new sequencing machines remains a challenge. While a plethora of tools are available to map the resulting reads to a reference genome, and to conduct primary analysis of the mappings, it is often necessary to visually examine the results and underlying data to confirm predictions and understand the functional effects, especially in the context of other datasets.

Results: We introduce Savant, the Sequence Annotation, Visualization and ANalysis Tool, a desktop visualization and analysis browser for genomic data. Savant was developed for visualizing and analyzing HTS data, with special care taken to enable dynamic visualization in the presence of gigabases of genomic reads and references the size of the human genome. Savant supports the visualization of genome-based sequence, point, interval, and continuous datasets, and multiple visualization modes that enable easy identification of genomic variants (including SNPs, structural, and copy-number variants), and functional genomic information (e.g. peaks in ChIP-seq data) in the context of genomic annotations.

Availability: Savant is freely available at http://compbio.cs.toronto.edu/savant

Contact: savant@cs.toronto.edu

1 INTRODUCTION

The emergence of High Throughput (aka Next-Generation) Sequencing technologies has made the acquisition of genomic data quicker and more affordable than ever before. Continued technological strides are being made to further improve throughput, cost, and accuracy of the sequencing platforms, enabling large-scale studies of genomes, populations, and diseases. Within the field of personalized genomics, ambitious sequencing projects such as the 1000 Genomes Project, the International Cancer Genome Consortium, and the Autism Genome Project, are seeking to identify genomic variants among human genomes and to use this knowledge to determine the genetic underpinnings of human diseases by associating variants with symptoms (Via *et al.*, 2010; http://www.icgc.org/; Hu-Lince *et al.*, 2005). Many computational tools have been developed for Single Nucleotide Polymorphism (SNP), Insertion/Deletion (indel), and other types of genetic variation discovery in individuals sequenced using High Throughput Sequencing (HTS) platforms (Li *et al.*, 2008; Chiang *et al.*, 2009; Hormozdiari *et al.*, 2009; see Dalca & Brudno, 2010 and Medvedev *et al.*, 2009 for reviews). The results of these analyses are being integrated into large scale datasets, annotating each variant with the allele frequency in various populations, and enabling demographic and association studies.

Simultaneously, HTS has also revolutionized functional genomics, where the ability to sequence RNA and DNA to extremely high coverage has made possible RNA-seq and ChIP-seq (Pepke *et al.*, 2009), methodologies that help discover rare RNA transcripts, identify the location of transcription factor binding sites on the genome, as well as discover the locations of nucleosomes. RNA-seq and ChIP-seq data also consist of many reads aligned to a reference genome, which are then binned and analyzed for peaks that indicate putative transcription factor binding sites and exons, respectively. Additionally, anomalously mapped reads or pairs from RNA-seq experiments can suggest alternative splicing or fusion transcripts.

The capacity of new sequencing technologies to generate huge volumes of raw sequence data has made its analysis a substantial informatics challenge. Sequencing machines typically output files containing a nucleotide sequence and quality values for each read, which can be tens to hundreds of gigabytes in size due to the sheer number of reads sequenced in a single run. These files can be easily parsed line-by-line by downstream computer programs because of their structured textual format, but are not intended for direct manual or visual analysis. While downstream analysis programs, such as read mappers, SNP callers, and peak finders can identify regions of interest to the bench scientist, a visual analysis of the supporting data, as well as other, orthogonal, information is usually warranted before costly wetlab experiments are performed to confirm the biological validity of the prediction.

Visualization of genomic data gives researchers the benefit of looking at information in a more natural and interpretable way compared to a textual representation. There are many different tasks that are facilitated by visualization including: (1) Integration of multiple related datasets into a single view, to gain insight into the interaction between genomic features, (2) Algorithm development, where visualization of many putative calls (e.g. genomic variants, promoter sites, intron-exon boundaries, etc.) helps with debugging

*to whom correspondence should be addressed

and identification of true and false positives, and (3) Exploration of various genomic regions for specific signatures of functional sites that may be difficult to describe within a computer program, e.g. two closely spaced peaks in ChIP-seq data indicating adjacent binding sites. Without the convenience of a visualization tool, in each of these settings all regions of interest would have to be painstakingly considered via manual analysis of the supporting data.

A number of tools have been developed for visualizing genome-based annotations. The UCSC and Ensembl genome browsers, for example, are popular online tools that have traditionally been used to display various biological datasets, such as genomic variants, ESTs, and functional genomic data, in the context of high-quality, manually curated annotations (Kent *et al.*, 2002; Hubbard *et al.*, 2009). While both have been recently updated to support display of HTS data, their use for this purpose is onerous because: (1) a large amount of locally stored data must be uploaded to servers across the internet, (2) once uploaded, the data being visualized cannot easily be manipulated or computed with, and (3) server-side browsers are typically slow and non-interactive. Other visualization programs like the Integrative Genomics Viewer, Artemis, and Tablet (http://www.broadinstitute.org/igv; Rutherford *et al.*, 2000; Milne *et al.*, 2010) are designed to run on conventional desktop computers and thus make use of local storage capacity and computing resources to overcome the shortcomings of web-based browsers. While these popular browsers allow for interactive visualization of HTS data, they have limited analytic capability, and are not extensible through user-contributed modules. A substantial barrier for researchers who use genomic visualization tools, for all types of data, is the disconnect between the processes of visualization and compute-intensive analyses (Nielsen *et al.*, 2010), a void which is caused by visualization tools being programmatically inaccessible.

Here we introduce Savant, the Sequence Annotation Visualization and ANalysis Tool, which combines visualization of HTS and other genome-based data with powerful analytic tools. The Savant feature set (summarized in Table 1) was guided by three key design principles. (1) Ease of use. Users can easily install the application, obtain and load data, and navigate to specific regions of interest. The general layout of data mirrors the standard genome browsers to shorten the learning curve. (2) Speed and efficiency. The program quickly and dynamically sifts through very large data sets while maintaining a reasonable memory footprint. (3) Access and extensibility. The underlying data is readily accessible from within the tool itself, and users can extend the application by adding any number of plug-ins for specific tasks.

Table 1. Savant Feature List

Feature categories	Features
Data Formats	FASTA, BED, SAM/BAM, WIG, GFF, and any tab delimited text file containing positional annotations
Speed and Efficiency	Fast data access Scalable to very large input files Small memory footprint
Navigation	Zooming, panning, and seeking using range controls or keyboard and mouse
Layout	Dockable modules supporting show/hide, rearrangement, maximize, float, close
Visualization	Compact view of sequence, point, interval, and continuous tracks Multiple display modes for specific track types
Language	Implemented in Java
Operating Systems	Works on Windows, Linux, Mac, and any other platform having support for Java Virtual Machine
Extensibility	Plugin framework allows access to data, tracks, and UI for extensive customization
Others	Bookmarking favorite locations Locking overview tracks Novel representation of paired-end reads making structural variations easy to identify

2 INTERFACE

Savant has a simple and intuitive interface, which is customizable through the use of a modular docking framework. Figure 1 shows a screenshot of Savant, and illustrates its various components, expanded upon in the following subsections.

Navigation

There are several ways a user can specify the genomic region to be displayed by the viewer. Coarse navigation is made possible through a range selection panel whose horizontal length represents the length of the loaded genome, from which subranges can be chosen using the mouse. Alternatively, fine navigation is possible by entering the desired genomic range into text fields. Neighboring buttons enable zooming in and out and panning left and right. Each of these functions can also be engaged through mouse and keyboard shortcuts. Savant also uses a bookmarking framework to allow the user to switch between many regions of interest, as described lower.

Modules and Docking Framework

Savant uses a modular docking framework, similar to those used in most Integrated Development Environments (IDEs). Each module within the application appears as a separate window that can be shown, hidden, maximized, minimized, resized, closed, or rearranged in any configuration the user desires. Modules can also be detached from the main interface and moved to a separate location, which is useful for maximizing screen usage on setups having multiple displays. Savant includes a number of modules which are described in subsections that follow. To demonstrate the power and utility of Savant's plug-in framework the Table View module was implemented as a plug-in.

2.1 Tracks

A track is a visual representation of one data set. Each track shows data of a single type, such as a genome, read alignment, gene set, or generic annotation. A user can specify the region of the loaded tracks to be displayed via the browser's many navigation controls. By default, multiple tracks are stacked on top of each other so that positions along their horizontal axes correspond to the same location of the genome, which is standard for genomic viewers (Staden *et al.*, 2000, Gordon *et al.*, 1998). Simultaneously, Savant enables the user to lock a certain track, while updating the others, as is described in Section 3.1

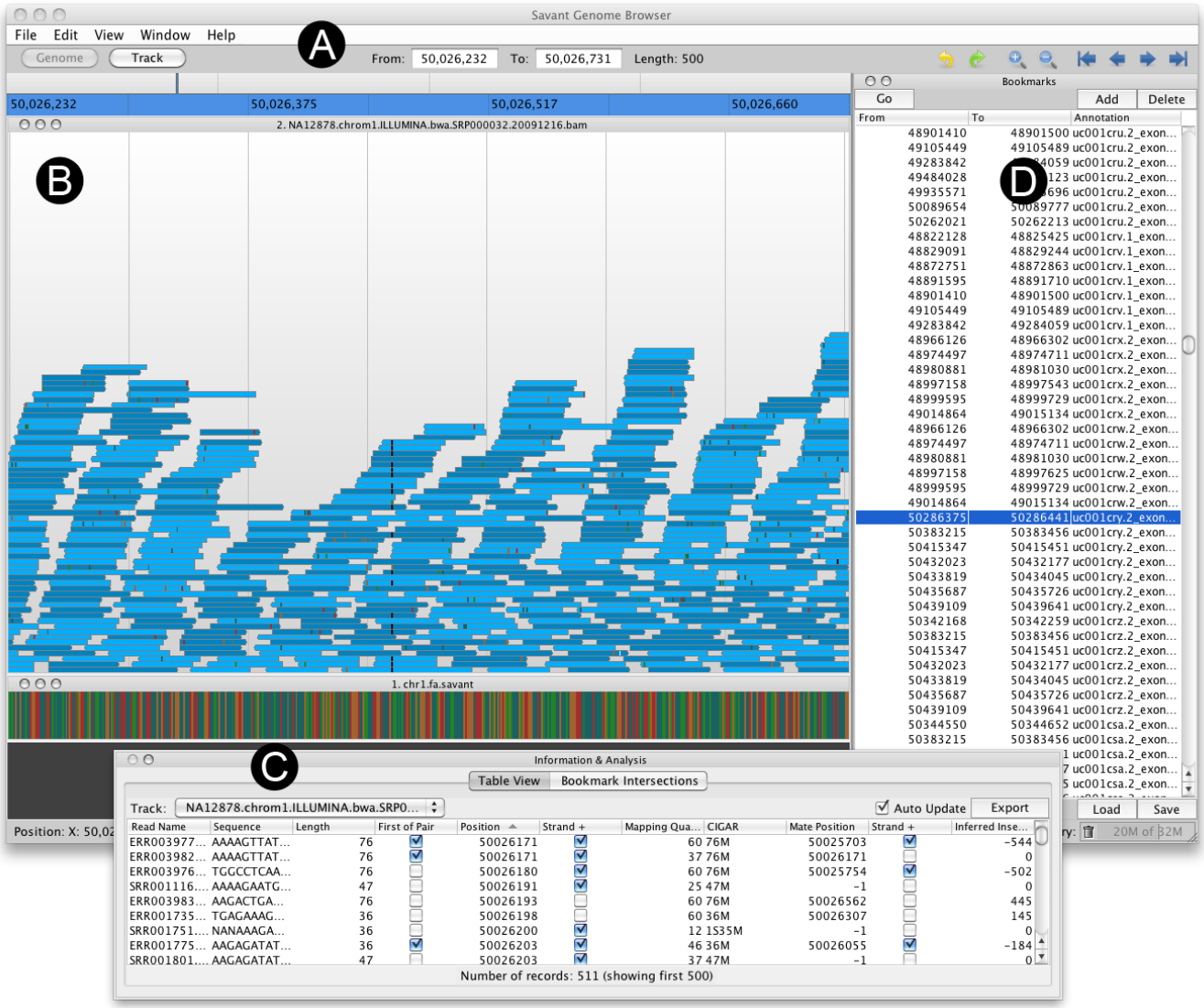

Fig. 1. Screenshot of Savant. A. Range controls. Selection, zoom, and pan controls for coarse navigation; text fields for fine navigation. Zooming and panning are also possible via keyboard and mouse commands. B. Tracks. These represent the data in current range. Top: read alignments, with colored pixels representing differences between the reads and the reference. Bottom: color representation of the genome sequence. C. Table View module, detached from the main interface. The table view module is displaying the mapped reads with SAM format fields. D. Bookmarks module.

2.2 Bookmarks

It is often useful to make note of interesting regions while using the browser, or to load a set of such regions in order to quickly navigate between them. The Bookmarks module helps keep track of such locations. Users may add, remove, or seek to a bookmarked region by using buttons within the module or keyboard shortcuts. Furthermore, bookmarks may be tagged with a description and exported for future use or for sharing among colleagues.

2.3 Table View

Finally, while Savant aims to provide the user with the ability to compute on the underlying data directly, through the plug-in framework described in Section 5, in many cases the user may wish to identify the underlying data elements for export to an external program, e.g. identifying the genomic sequence within a window to align against another genome, or downloading all of the supporting reads for a SNP to make sure they do not align elsewhere in the genome. In Savant the user can display the underlying textual data from any loaded track within the Table View. This module displays records as rows and fields as columns in a spreadsheet. For each read mapping, for example, the Table View displays the read name, mapped position, CIGAR string, and other SAM fields. The data can be sorted in either ascending or descending order with respect to any field. The spreadsheet can also be exported for further analyses.

3 VISUALIZATION

Savant retrieves and renders data every time a range change is requested by the user. Together, these processes happen quickly so as to confer seamless navigation around the genome. The renderer for each track is adaptive to both the display mode and the length of the viewed region chosen by the user.

3.1 Dynamic Resolution Changes

Savant dynamically adjusts its resolution – the amount of information it displays – to optimize both nucleotide- and genome-scale visualization of tracks. For example read alignments can be visualized as a coverage track when the number of base pairs within a region is too large to enable the visualization of individual reads. Once the region is small enough, Savant seamlessly switches to a read-alignment view, as shown in Figure 2. In addition to presenting a more intuitive visualization, this feature also reduces the program's memory footprint and improves overall speed.

Individual tracks can also be locked to a particular range so that they are not updated until they are unlocked. Locked tracks can be used as overview profiles from which subregions can be selected to specify range changes for other tracks. Track locking also enables simultaneous viewing of high- and low-resolution profiles.

3.2 Display Modes

Particular data types can be displayed in different modes. For example, interval annotations can be squished together on a single line or packed neatly so that none overlap (mimicking the squish and pack modes of the UCSC browser). The representation can be dynamically toggled within the browser, with each representation option emphasizing different aspects of the data. The variant and strand modes for read alignments, for instance, use colors to emphasize mismatches in reads and the strands to which reads are mapped, respectively. A novel mode for representing matepairs shows arcs between the mapped locations of paired reads, where the height of each arc is proportional to the inferred insert size. Arcs for anomalously mapped pairs, such as those suggestive of inversions or duplications, are colored differently. The various modes for read alignments are illustrated in Figure 2.

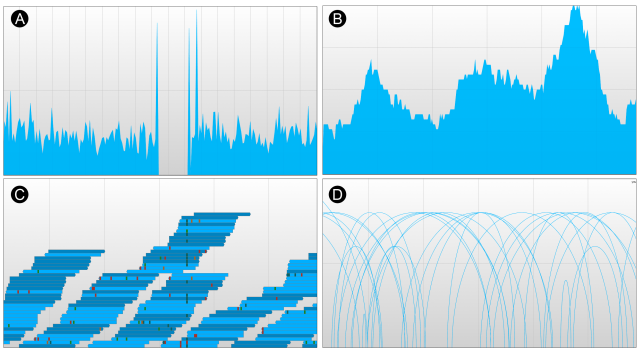

Fig. 2. Read alignments, visualized at various resolutions and using two modes. A: Chromosome-wide view of read mappings, showing the overall coverage (with no coverage in the centromere). B: Regional view, still visualized as a coverage map, showing higher coverage in certain regions of the genome. C: Local view, the reads are shown separately and differences between the reads and the reference genome are colored. Reads on the forward and reverse strand are shown with different shades of blue. D: Matepair (arc) mode, showing the relative distance between the two reads of a pair. Taller arcs indicate larger distances between the pairs.

Table 2. Supported File Formats

File Format	Description
FASTA	Standard format for nucleotide sequences
BED	Format for describing co-ordinates of localized features on genomes, such as genes
SAM/BAM	Relatively standard format for large sets of nucleotide sequence alignments
WIG	Standard format for continuous-valued data. Useful for GC percent, probability scores, and transcriptome data
GFF	General feature format for annotations of ranges
Tab-delimited	Any tab-delimited file containing point, interval, or continuous genome annotations

4 FILE FORMATTING AND PERFORMANCE

Savant supports a number of common text-based formats, which are described in Table 2. However, because text-files do not enable fast random access, Savant formats and saves each file so as to provide very efficient search operations at runtime. The speed with which Savant can sift through large datasets is enabled by the way in which it formats and indexes data. In particular, formatting involves converting text records into an indexed binary data structure specific to each data type. Sequence and continuous tracks are stored as fixed-width records, enabling direct lookup of records of interest. Annotation ranges (such as genes) are stored using a binning scheme similar to the one used in the UCSC Browser (Kent *et al.*, 2002) and in BAM files (Li *et al.*, 2009), so retrieving all ranges corresponding to some region usually requires only one, and at most $O(logn)$ disk seeks. File formatting can be done directly through the application itself. Savant keeps its memory usage low by adjusting its sampling rate depending on the size of the visualized range.

The time and space requirements for the processes of data formatting and visualization were measured for a collection of human chromosome 1 datasets, including a genetic sequence, genes, SNPs, mammalian conservation, and alignments of sequenced reads from an individual from the 1000 Genomes Project (∼40X coverage). The tests were performed on a Lenovo T61p laptop computer with an Intel Core 2 Duo CPU at 2.40GHz and 3.0 GB of RAM. The results are summarized in Table 3. Formatting of the gene, SNP, sequence, and conservation tracks took less than 10 minutes total, while the computation of a coverage track from a large set of read alignments took about an additional 40 minutes. The latter conversion is optional, and allows for dynamic switching between an alignmnent view and a coverage view as illustrated in Figure 2. Runtime performance was assessed by measuring the time taken to navigate to ranges of various sizes . Each measurement was performed on a newly started instance of Savant and the start location of the range was randomized. For seeking to arbitrary

ranges of sizes ten million – 10k basepairs, Savant took less than a second to fetch and render data. The performance was worst for ranges just slightly shorter than 10k long, where the large number of BAM records that are displayed require 2 seconds to be fetched from disk. Savant renders virtually instantaneously for regions having sizes on the order of hundreds of basepairs, where most fine-scale visualization is done.

5 PLUG-IN FRAMEWORK

Savant is able to integrate user-defined plug-ins, allowing one to accomplish very specific tasks. Each plug-in can be one of two general types. Interactive plug-ins are allocated dockable modules on which GUI elements such as buttons or text fields can be placed to respond to user input and retrieve data. Non-interactive plug-ins are not designated a GUI component but still have extensive access to the innards of the browser. A number of helper functions are provided to plug-ins which are summarized in Table 4. Plug-ins can be used, for example, to prototype a SNP-finder by identifying variable columns currently in view, or for computing genome-wide statistics, such as the fraction of SNPs in exons.

Plugin development is straight-forward and requires implementation of one Java interface. The Bookmark Intersection Plug-in, shown in Figure 3, is an example of an interactive plug-in. The plug-in allows the user to select two tracks, intersect them, and load the intersecting regions into the list of bookmarks, enabling easy navigation to all of the regions of interest. Once developed, plug-ins can be shared among users through the Plugin section of the Savant website.

```
public class BookmarkIntersectionPlugin
  implements InteractivePlugin {

  // set by UI
  GenericIntervalTrack sourceTrack, targetTrack;

  RangeController rc;
  BookmarkController bc;

  // constructor
  public BookmarkIntersectionPlugin(JPanel p,
      SavantPluginAdapter pluginAdapter) {
   /* UI Setup Code:
    * Create a UI in JPanel to select a src and a
    * target track. Details skipped for simplicity
    */
    rc = pluginAdapter.getRangeController();
    bc = pluginAdapter.getBookmarkController();
  }

  /* called on button press */
  public static void bookMarkTrackIntersections() {

    List sourceIntervals =
        sourceTrack.getRecords(rc.getMaxRange());

    for (GenericIntervalRecord r : sourceIntervals) {
      Range sourceRange = r.getInterval().getRange();

      List targetIntervals =
        targetTrack.getRecords(sourceRange);

      if (targetIntervals.size() > 0) {
        bc.addBookmark(new Bookmark(sourceRange,
          targetIntervals.getDescription()));
      }
    }
  }
}
```

Fig. 3. Code used to make Bookmark Intersection Plug-in. The details of the UI that allows the user to select two tracks have been omitted. Once the two tracks are selected, the bookMarkTrackIntersections() method is run, which, for each interval of one track, finds overlapping intervals of the other, and saves intervals with overlap to the bookmark panel.

6 EXAMPLE USES

In this section we demonstrate example usages of Savant to identify and visualize various polymorphisms between the reference human genome and a Yoruban genome (HapMap individual NA18507) sequenced with the Illumina platform (Bentley *et al.*, 2008). The raw reads and MAQ mappings (in BAM format) were downloaded from the 1000 Genomes website, while the conservation track (in WIG format), gene annotations, and dbSNP data were obtained from the UCSC database. Figure 4A shows two likely SNP variants in the NA18507 genome. SNPs are typically supported by consistent mismatches in aligned reads with respect to the reference at the variant positions. The reads support the existence of a transition mutation (C to T) at a known SNP site, indicating a likely common variant in the human population. There is also considerable evidence (3/12 reads) for another transition (G to A) slightly upstream, indicating a putatively novel SNP. Both of these SNPs fall in an exon of a gene, as shown by the annotation, while the conservation appears to drop-off in the immediate vicinity of the SNPs, indicating weakened evolutionary constraint.

We also demonstrate the use of the plug-in framework to identify and investigate biologically interesting indel predictions made using an external tool. We used the dataset of predicted insertion and deletion (indel) polymorphisms predicted by MoDIL (Lee *et al.*, 2009), which identifies these variants by analysis of matepair data. In particular, MoDIL identifies regions of the genome with a significant number of discordant matepairs (those with a distance between the mappings of the reads significantly different from the expected insert size of the library). We built a plug-in to identify those indels that overlap exons, the code of which is shown in Figure 3. Figure 4B displays the result of running this plug-in, together with one such region of interest. In this case, the deletion lies directly between two exons, and likely creates a genomic variant with a much shorter intron, or with the intron completely spliced out of the gene sequence. Taller arcs in the read visualization mode are indicative of ”stretched” matepairs that confirm the deletion, while the lack of read mappings within the deleted region indicates that the deletion is homozygous.

Table 4. Helper functions provided to plug-ins

Category	Function descriptions
Range	change range
Tracks	add, remove, retrieve data, change display modes and resolutions
Bookmarks	add, remove, and seek
UI	rearrange modules
Other	take screenshot, export data, etc.

DATA FILES			FORMATTING		RETRIEVAL AND VISUALIZATION							
					10M		100K		1K		100	
File name	Size	Records	Out Size	Time	#Recs.	Time (s)	#Recs.	Time (s)	#Recs.	Time (s)	#Recs.	Time (s)
chr1.genes.bed	1.1 MB	7.5K genes	970 KB	2s	423	0.18	10	0.05	7	0.03	5	0.00
chr1.snps.point	26 MB	1.5M SNPs	26 MB	4s	–	–	559	0.01	12	0.00	6	0.00
chr1.sequence.fasta	242 MB	249M nucs	237 MB	11s	–	–	–	–	1,000	0.00	100	0.00
chr1.conservation.wig	1.2 GB	249M vals	1.2 GB	6m	10M*	0.32	100K*	0.18	1,000	0.02	100	0.01
NA12878.chr1...bam	25 GB	262M reads	No formatting req.		Coverage shown at these ranges				1,101	0.05	150	0.04
NA12878.chr1...cov	Computed from BAM file		1.2 GB	42m	10M*	0.32	100K*	0.18	Read alignments shown at these ranges			
TOTAL				**53m**		**0.82s**		**0.40s**		**0.10s**		**0.05s**

Table 3. Time and space requirements for file formatting and visualization. The Data Files section describes input files. The Formatting section shows time required to format each input file and the resulting formatted file size. The Retrieval and Visualization section shows the time taken to retrieve data from ranges of various sizes and the number of records retrieved and drawn. All operations require less than 50 MB of memory. Tracks with continuous values are smoothed before rendering at large ranges, denoted with an asterisk (*). Sequence and point tracks are not rendered beyond certain ranges, denoted with a hyphen (–). Read alignments (BAM file) are replaced by coverage (pre-computed from the BAM file) when visualizing longer regions.

7 DISCUSSION AND FUTURE WORK

In this paper we introduce Savant, a genome browser customized for visualization of HTS data. In addition to dynamic visualization and support for multiple data types, Savant supports multiple novel features meant to simplify the analysis of genomic data, including a Bookmarks module to keep track of regions of interest, a Table View module to allow researchers access to the raw data underlying the visualizations, and a plug-in framework that allows developers extensive access to the browser, enabling them to program novel analysis methods and extend the browser to support new visualization and data types.

In the near future we plan to expand Savant by supporting additional file formats, while also allowing users to automatically download annotation tracks from various public resources, such as the UCSC Genome Browser and the 1000 Genomes Project. Additional functionality, for example to save/restore user sessions, enabling novel visualization modes, such as superimposition of tracks, and combining the matepair and regular read alignment views in a single panel will also be added in subsequent releases. Savant is an open-source tool, freely available at http://compbio.cs.toronto.edu/savant/. We are also working to build a user community, accessible through the same website. The community can be used to share Savant plug-ins with other researchers, as well as to communicate with the development team, to report bugs and to suggest new functionality and improvements.

ACKNOWLEDGEMENTS

We would like to thank Misko Dzamba, Seunghak Lee, and Paul Medvedev for assistance with this manuscript. We are also indebted to the scientists at The Centre for Applied Genomics (Toronto, Canada) for feature and design suggestions. This work was supported by a CIHR Catalyst Grant to MB and an NSERC USRA for MF. MB is an Alfred P. Sloan Research Fellow.

REFERENCES

Bentley, D. R. *et al.* (2008) Accurate whole human genome sequencing using reversible terminator chemistry. *Nature,* **456** (7218), 53–59.

Chiang, D. Y. *et al.* (2009) High-resolution mapping of copy-number alterations with massively parallel sequencing. *Nat. Methods,* **6** (1), 99–103.

Dalca, A. V. & Brudno, M. (2010) Genome variation discovery with high-throughput sequencing data. *Brief Bioinform,* **11** (1), 3–14.

Gordon, D. *et al.* (1998) Consed: a graphical tool for sequence finishing. *Genome Research,* **8** (3), 195–202.

Hormozdiari, F. *et al.* (2009) Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome Res.,* **19** (7), 1270–1278.

Hu-Lince, D. *et al.* (2005) The autism genome project: goals and strategies. *Am. J. Pharmacogenomics,* **5** (4), 233–246.

Hubbard, T. J. P. *et al.* (2009) Ensembl 2009. *Nucl. Acids Res.,* **37** (suppl_1), D690–697.

Kent, W. J. *et al.* (2002) The human genome browser at ucsc. *Genome Res.,* **12** (6), 996–1006.

Lee, S. *et al.* (2009) Modil: detecting small indels from clone-end sequencing with mixtures of distributions. *Nat. Methods,* **6** (7), 473–474.

Li, H. *et al.* (2008) Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research,* **18** (11), 1851–1858.

Li, H. *et al.* (2009) The sequence alignment/map format and samtools. *Bioinformatics (Oxford, England),* **25** (16), 2078–2079.

Medvedev, P. *et al.* (2009) Computational methods for discovering structural variation with next-generation sequencing. *Nature Methods,* **6** (11s), S13–S20.

Milne, I. *et al.* (2010) Tablet–next generation sequence assembly visualization. *Bioinformatics,* **26** (3), 401–402.

Nielsen, C. B. *et al.* (2010) Visualizing genomes: techniques and challenges. *Nature methods,* **7** (3 Suppl).

Pepke, S. *et al.* (2009) Computation for chip-seq and rna-seq studies. *Nat. methods,* **6** (11 Suppl), S22–S32.

Rutherford, K. *et al.* (2000) Artemis: sequence visualization and annotation. *Bioinformatics,* **16** (10), 944–945.

Staden, R. *et al.* (2000) The staden package, 1998. *Methods in molecular biology (Clifton, N.J.),* **132**, 115–130.

Via, M. *et al.* (2010) The 1000 genomes project: new opportunities for research and social challenges. *Genome Med.,* **2** (1), 3+.

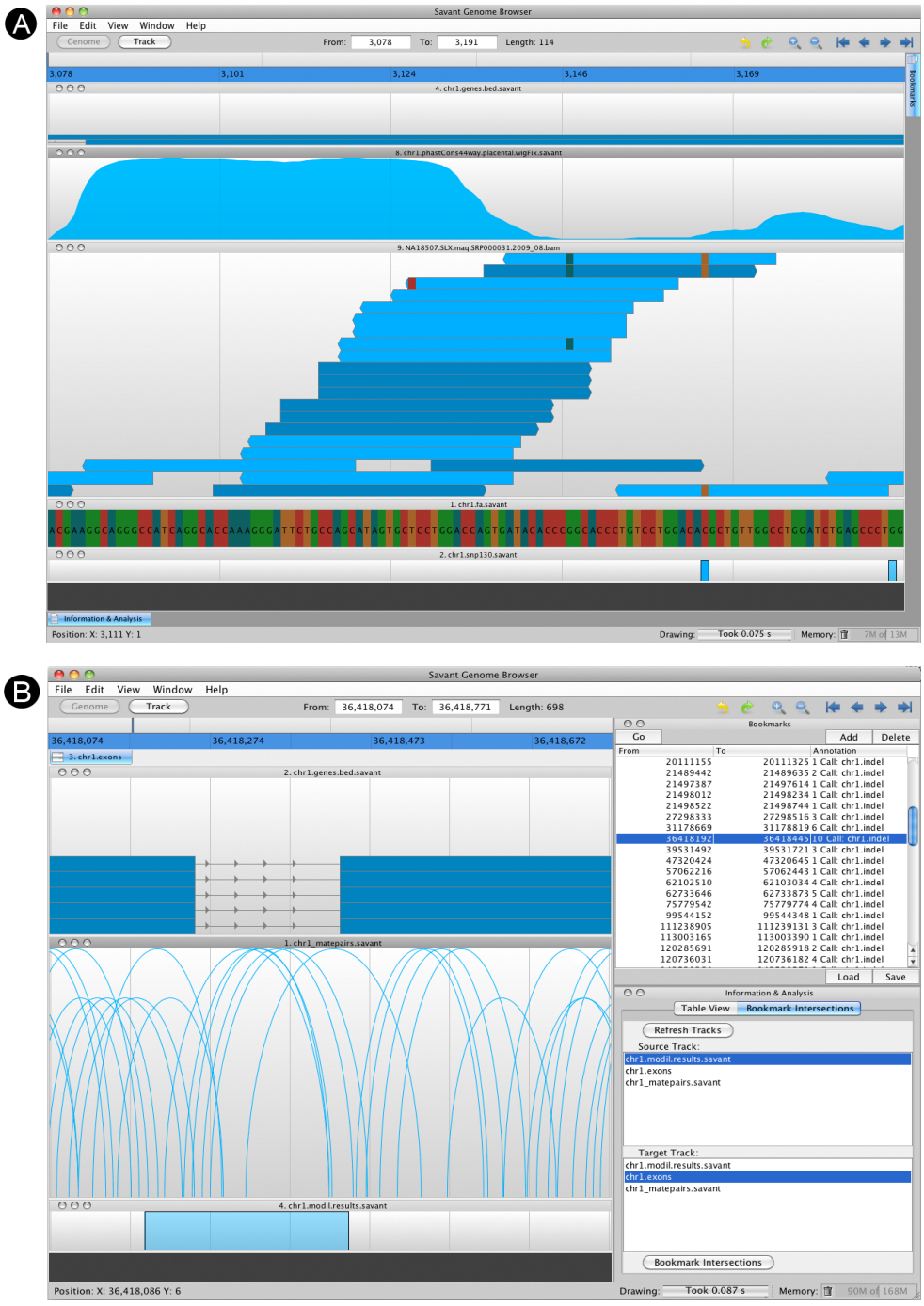

Fig. 4. A. Visualization of two SNP variants. The displayed tracks are, top to bottom, Conservation, Gene Models, Read Alignments, the Genome sequence, and known SNPs from dbSNP. Two potential SNP variants are indicated by consistent mismatching colors within a column, with the downstream SNP previously known, while the upstream one, a heterozygous variant, not in dbSNP. B. Identification and visualization of MoDIL indel variants that overlap exons via the plug-in framework. The visualized tracks are Gene Models, read mappings (visualized in the matepair mode), and MoDIL predictions. The panel on top right is the Bookmark module, while on the bottom right is the BookMarkIntersection plug-in described in Figure 3. To identify variants of interest, the user selects the two tracks in the BookMarkIntersection window, and those variants that overlap exons are added to the list of bookmarks. The user can then easily go through this list. In this particular case the indel likely occurs in the intron between the exons, but because of MoDIL's inability to accurately identify borders of variants, the variant is shown as overlapping.